

# EFFICIENT MONTE CARLO OPTIMIZATION FOR MULTI-DIMENSIONAL CLASSIFIER CHAINS

Jesse Read, Luca Martino\*

Dept. of Signal Theory and Communications  
Universidad Carlos III  
Madrid 28911, Spain.

David Luengo\*

Dept. of Circuits and Systems Engineering  
Universidad Politécnica de Madrid  
Madrid 28031, Spain.

## ABSTRACT

Multi-dimensional classification (MDC) is the supervised learning problem where an instance may be associated with multiple classes, rather than with a single class as in traditional binary or multi-class single-dimensional classification (SDC) problems. MDC is closely related to multi-task learning, and multi-target learning (generally, in the literature, multi-target refers to the regression case). Modeling dependencies between labels allows MDC methods to improve their performance at the expense of an increased computational cost. In this paper we focus on the classifier chains (CC) approach for modeling dependencies. On the one hand, the original CC algorithm makes a greedy approximation, and is fast but tends to propagate errors down the chain. On the other hand, a recent Bayes-optimal method improves the performance, but is computationally intractable in practice. Here we present novel Monte Carlo schemes, both for finding a good chain sequence and performing efficient inference. Our algorithms remain tractable for high-dimensional data sets and obtains the best overall accuracy, as shown on several real data sets.

**Index Terms**— multi-label; classification; Monte Carlo; chaining

## 1. INTRODUCTION

Multi-dimensional classification (MDC) is the supervised learning problem where an instance may be associated with multiple classes, rather than with a single class as in traditional binary or multi-class single-dimensional classification (SDC) problems. MDC is closely related to multi-task learning, and multi-target learning (generally, in the literature, multi-target refers to the regression case). The recently popularised task of multi-label classification (see [1, 2] for

overviews) can be viewed as a particular case of the multi-dimensional problem that only involves binary classes, considered as *labels* that can be turned on (1) or off (0) for any data instance.

The MDC learning context is receiving increased attention in the literature, since it arises naturally in a wide variety of domains such as text, audio, still images and video, bioinformatics, medical diagnoses [2, 1, 3]. The main challenge in this area is modeling label dependencies without incurring in an intractable complexity.

A basic approach to MLC is to the independent classifiers (**IC**) method, which decomposes the MDC problem into a set of SDC problems (one per label) and uses a separate classifier for each label variable<sup>1</sup>. In this way, MDC is turned into a series of standard SDC problems that can be solved with any off-the-shelf binary classifier (e.g., a logistic regressor or a support vector machine<sup>2</sup>). Unfortunately, although **IC** has a low computational cost, it cannot provide high performance, because it does not model dependencies between labels [1, 5, 6, 7, 8, 3].

In order to model dependencies explicitly, several alternative schemes have been proposed, such as the so-called *label powerset* (**LP**) method [9]. **LP** considers each potential combination of labels in the MLC problem as a single label. In this way, the multi-label problem is turned into a traditional multi-class problem that can be solved using standard methods. Unfortunately, given the huge number of class values produced by this transformation, this method is usually unfeasible for practical application, and suffers from issues like overfitting. This was recognised by [5, 10], which provide approximations to the **LP** scheme that reduce these problems, although such methods have been superseded in recent years.

A more recent idea is using classifier chains (CC), which improves the performance of **IC** and **LP** by constructing a sequence of classifiers that make use of previous outputs of the chain. The original CC method, introduced in [6] and ex-

\*Supported by funding from Ministerio de Ciencia e Innovación of Spain (project MONIN, ref. TEC-2006-13514-C02- 01/TCM, Program Consolider-Ingenio 2010, ref. CSD2008- 00010 COMONSENS, and Distributed Learning Communication and Information Processing (DEIPRO) ref. TEC2009-14504-C02-01) and Comunidad Autónoma de Madrid (project PROMULTIDIS-CM, ref. S-0505/TIC/0233).

<sup>1</sup>We henceforth use the term *label* throughout to refer generally to a *class variable*; and not necessarily a binary-only tag, as is the case in much of the multi-label literature.

<sup>2</sup>support vector machines are naturally binary, but can easily be adapted to multi-class by using a pairwise voting scheme, as in [4]

tended in [7, 3], makes a greedy approximation, and is fast but tends to propagate errors down the chain. Nevertheless, a very recent extensive experimental comparison reaffirmed that CC is among the highest-performing methods for MLC, and recommended it as a benchmark algorithm [11]. A CC-based Bayes-optimal method, probabilistic classifier chains (PCC), has also been recently proposed [7]. However, although it improves the performance of CC, its computational cost is too large for most real-world applications.

In this paper we introduce a different novel methods that attains the performance of PCC, but remains tractable for high-dimensional data sets. Our approaches are based on a double Monte Carlo optimization technique and, unlike all other chain-based methods in the literature, it explicitly searches the space of possible chain-sequences during the training stage. Hence, predictive performance can be traded off for scalability depending on the application.

## 2. MULTI-DIMENSIONAL CLASSIFICATION (MDC)

Let us assume that we have a set of training data composed of  $N$  labelled examples,  $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ , where  $\mathbf{x}^{(i)} = [x_1^{(i)}, \dots, x_D^{(i)}]^\top$  is the  $i$ -th  $D$ -dimensional instance (input), with  $x_d^{(i)} \in \mathcal{X}_d$  for  $1 \leq d \leq D$ , and  $\mathbf{y}^{(i)} = [y_1^{(i)}, \dots, y_L^{(i)}]^\top$  is the  $i$ -th example's  $L \times 1$  label relevance vector (output), with

$$y_j^{(i)} \in \{1, \dots, K_j\} = \mathcal{Y}_j,$$

where  $K_j$  is a finite number of classes, being its  $j$ -th class assignment. In MDC we seek to learn a function,  $\mathbf{y} = \mathbf{h}(\mathbf{x})$ , that assigns a vector of labels,

$$\mathbf{y} \in \mathcal{Y}_1 \times \dots \times \mathcal{Y}_L,$$

to each instance,

$$\mathbf{x} \in \mathcal{X}_1 \times \dots \times \mathcal{X}_d \subseteq \mathbb{R}^d.$$

Let us assume that the *true* posterior distribution of the data is  $f(\mathbf{y}|\mathbf{x})$ . From a Bayesian point of view, the optimal label assignment for a given test instance,  $\mathbf{x}^*$ , is provided by the maximum a posteriori (MAP) label estimate:

$$\mathbf{y}_{\text{MAP}}^* = \mathbf{h}_{\text{MAP}}(\mathbf{x}^*) = \underset{\mathbf{y}}{\operatorname{argmax}} f(\mathbf{y}|\mathbf{x}^*). \quad (1)$$

Unfortunately, the true distribution,  $f(\mathbf{y}|\mathbf{x})$ , is usually unknown, and the classifier has to work with an approximation,  $p(\mathbf{y}|\mathbf{x})$ , constructed from the training data. Hence, the (possibly sub-optimal) label prediction is finally given by

$$\mathbf{y}_{\text{MAP}}^* \approx \mathbf{y}^* = \mathbf{h}(\mathbf{x}^*) = \underset{\mathbf{y}}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{x}^*). \quad (2)$$

Although binary-only multi-label problems can be considered as a subset of multi-dimensional problems, the reverse is not true, and there are some crucial differences meaning

that much research in MLC is not directly applicable to MDC. In the first instance, there is a higher dimensionality for the same  $L$ ; MLC deals with  $2^L$  possible values, and MDC deals with  $\prod_{j=1}^L K_j$ . Moreover, in MDC there are not only more possible, but many more *likely* classifications, due to a qualitative difference between tagging data with binary labels, and assigning classes, even if these classes are binary classes. In typical MLC problems, the binary classes are used to indicate *relevance* (e.g., 1) and irrelevance (e.g., 0). For example, the label “beach” may be relevant to a particular picture. In practice, on average typically only slightly more than  $1/L$  labels are relevant to each example [1] (see also Table 2). This also means that, the chance of  $\ell$  labels being relevant to a single data instance falls to zero as  $\ell \rightarrow L$  (it is very reasonable to expect that no data instance will be assigned all – or even a majority – of labels). This means that we have more prior information. In MDC however, classes (including binary classes) are used differently, for example indicating “male”/“female”. Clearly (prior-knowledge of the problem aside), we expect the chance of a particular data instance being classified “male” to be around 0.5. In summary, in MDC, the *practical*  $\mathbf{y}$ -space is much greater than in MLC, making probabilistic inference more challenging.

### 2.1. Independent Classifiers (IC)

Using independent classifiers (IC) is commonly mentioned in the MLC and MDC literature [9, 2, 6, 3]. This approach transforms the multi-dimensional problem into  $L$  separate uni-dimensional (i.e., standard binary or multi-class) problems. Hence for each  $j = 1, \dots, L$  a classifier  $h_j$  is employed to map new data instances to the relevance of the  $j$ -th label, i.e.,

$$\mathbf{y}_{\text{MAP}}^* \approx \mathbf{y}^* = \mathbf{h}(\mathbf{x}^*) = [h_1(\mathbf{x}^*), \dots, h_L(\mathbf{x}^*)],$$

where, probabilistically speaking, we can define for each  $h_j$  as

$$y_j^* = h_j(\mathbf{x}^*) := \underset{y_j}{\operatorname{argmax}} p(y_j|\mathbf{x}^*). \quad (3)$$

As we remarked in Section 1, this method is easy to build using off-the-shelf classifiers but it does not explicitly model label dependencies, and its performance suffers as a result. In fact, it assumes complete independence, i.e., that

$$p(\mathbf{y}|\mathbf{x}^*) = \prod_{j=1}^L p(y_j|\mathbf{x}^*). \quad (4)$$

We always expect some label dependencies in a multi-label problem (otherwise we are simply dealing with a collection of unrelated problems); some labels occur more likely together, or mutually exclusively. It is important to model these dependencies, because doing so can influence the outcome of predictions.

## 2.2. Classifier Chains (CC)

In [6], correlation among labels is considered. *Classifier chains* (CC) is based on modeling the correlation among labels using the chain rule of probability. Given a data instance,  $\mathbf{x}^*$ , and a vector of label indexes,  $\mathbf{s} = [s_1, \dots, s_L]^\top$ , obtained as a permutation of  $\{1, \dots, L\}$ ,  $p(\mathbf{y}|\mathbf{x}, \mathbf{s})$  may be expressed as<sup>3</sup>

$$p(\tilde{\mathbf{y}}|\mathbf{x}^*, \mathbf{s}) = p(\tilde{y}_1|\mathbf{x}^*) \prod_{j=2}^L p(\tilde{y}_j|\mathbf{x}^*, \tilde{y}_1, \dots, \tilde{y}_{j-1}), \quad (5)$$

where  $\tilde{\mathbf{y}} = [\tilde{y}_1, \dots, \tilde{y}_L]^\top$  is the permuted label vector (see Fig. 1), i.e.,  $\tilde{y}_j = y_{s_j}$  is the  $j$ -th label in the permutation,  $j = 1, \dots, L$ , and the probabilities in (5) are learnt from the labelled data during the training stage. It is important to remark that, theoretically Eq. (5) does not depend on the label order. However, since all the conditional densities in Eq. (5) are estimated from the training data, the label order can have a large effect in practice, as recognized by [7]. Given a data instance  $\mathbf{x}^*$  and a label order  $\mathbf{s}$ , note that the vector  $\tilde{\mathbf{y}}$  can be seen as a path in a tree with  $L$  levels, and  $p(\tilde{\mathbf{y}}|\mathbf{x}, \mathbf{s})$  is the “utility” corresponding to this path. Figure 2 depicts an example with  $K_j = 2$  for all  $j = 1, \dots, L$ .

First of all, CC considers arbitrarily a label order  $\mathbf{s}$ . Then during the test stage, given a instance  $\mathbf{x}^*$ , CC follows a single path of labels  $\tilde{\mathbf{y}}^*$  greedily down the chain of  $L$  binary classifiers, with the  $j$ -th classifier,  $h_j$ , predicting the  $j$ -th label’s relevance,  $\tilde{y}_j^*$ , using all previous predictions  $(\tilde{y}_1^*, \dots, \tilde{y}_{j-1}^*)$  as

$$\tilde{y}_j^* = h_j(\mathbf{x}^*|\mathbf{s}) = \operatorname{argmax}_{\tilde{y}_j} p(\tilde{y}_j|\mathbf{x}^*, \tilde{y}_1^*, \dots, \tilde{y}_{j-1}^*). \quad (6)$$

In carrying out classification down a chain in this way, CC models label dependencies and, as a result, usually performs much better than IC, while being similar in memory and time requirements in practice. However, due to its greedy approach (i.e., only one path is explored) and depending on the choice of  $\mathbf{s}$ , it is susceptible to errors in the initial links of the chain [7]. Figure 1 depicts an example of sequence of labels  $\mathbf{s}$  in the assumed correlation structure.

## 2.3. Probabilistic Classifier Chains (PCC)

*Probabilistic classifier chains* (PCC) was introduced in [7]. In the training phase, PCC is identical to CC; considering a particular order of labels  $\mathbf{s}$  (either chosen randomly, or as per default in the dataset). However, during the test stage PCC provides Bayes-optimal inference by exploring all the  $\prod_{j=1}^L K_j = 2^L$  possible paths (they consider the simplest case  $K_j = 2$  for all  $j = 1, \dots, L$ ). Hence, for a given test

<sup>3</sup>Note that, using the *true* conditional distributions, Eq. (5) does not depend on the label order. Namely, using the true densities Eq. (5) is always exact for any choice of  $\mathbf{s}$ .

instance,  $\mathbf{x}^*$ , PCC provides the optimum label estimate, obtained maximizing the label vector,  $\tilde{\mathbf{y}}$ , rather than the individual labels,  $\tilde{y}_j$ , i.e.,

$$\tilde{\mathbf{y}}^* = \mathbf{h}(\mathbf{x}^*|\mathbf{s}) = \operatorname{argmax}_{\tilde{\mathbf{y}}} p(\tilde{\mathbf{y}}|\mathbf{x}^*, \mathbf{s}), \quad (7)$$

where  $p(\tilde{\mathbf{y}}|\mathbf{x}^*, \mathbf{s})$  is given by (5). In [7] an overall improvement of PCC over CC is reported, but at the price of high computational complexity: it is intractable for more than about 10 labels ( $\equiv 2^{10}$  paths), which represents the majority of problems in the multi-label domain. Moreover, since all the conditional densities in (5) are estimated from the training data, the results can depend on the chosen label order  $\mathbf{s}$ .

## 2.4. Bayesian Network Classifiers

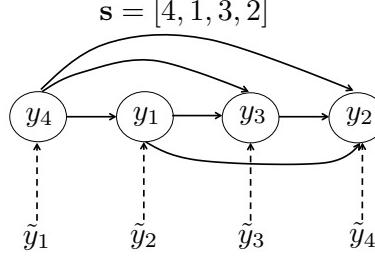
In [8] conditional dependency networks (CDN) are used as a way of avoiding choosing a specific label order  $\mathbf{s}$ . Whereas both CC and PCC are dependent on the order that labels appear in the chain, CDN avoids this problem: it is a fully connected network, rather than a chain. This fully connected network is comprised of  $L$  label-nodes  $p(y_j|\mathbf{x}, \mathbf{y}_{-j})$  for nodes  $j = 1, \dots, L$  (where  $\mathbf{y}_{-j} = [y_1, \dots, y_{j-1}, y_{j+1}, \dots, y_L]$  is the vector of all labels except the  $j$ th). In this method, Gibbs sampling is used for inference over  $I$  steps for collecting the marginal probabilities. Due to having  $(L(L - 1)/2)$  links inference may be problematic for large  $L$ .

[3] presents a more tractable network; finding an approximate representation of the dependency structure by using the Chow-Liu algorithm. They avoid needing a costly graph inference (as for example used in [8]’s CDNs) by treating the graph as  $L$  trees, where the  $j$ th node is the root node of the  $j$ th tree. However, this method is similar to CC in the sense that classification depends on the order of nodes. Unlike CC it does not model all dependencies (the dependence between leaf variables is not necessarily modelled).

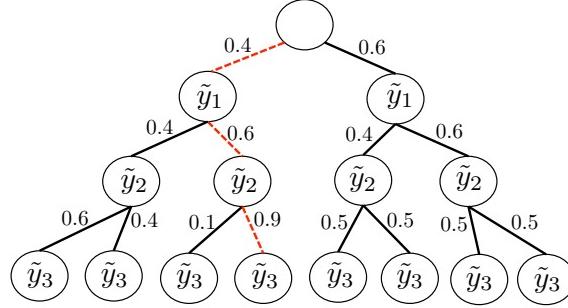
In the next section, we present a double Monte Carlo optimization technique to both find a good chain sequence and efficiently approximate Eq. (5) at inference time. The resulting method out-competes all the methods presented so far in this section without losing tractability to practical problems.

## 3. EFFICIENT DOUBLE MONTE CARLO TECHNIQUE FOR CLASSIFIER CHAINS

In chain-based MDC problems, for any given test instance,  $\mathbf{x}^*$ , and label order,  $\mathbf{s}$ , we wish to find the best label-relevance vector,  $\tilde{\mathbf{y}}^* = [\tilde{y}_1^*, \dots, \tilde{y}_L^*]$ , out of the  $\prod_{j=1}^L K_j$  possible label vectors (different paths). However, the best inference on a poor model will not be as good as the best inference on a good model. Therefore, at training time we also wish to find the best chain order or *sequence*,  $\mathbf{s} = [s_1, \dots, s_L]$ , out of the  $L!$  possible chains.



**Fig. 1.** Example of correlation structure in a classifier chain with  $L = 4$ . In the example, we have  $\mathbf{s} = [s_1 = 4, s_2 = 1, s_3 = 3, s_4 = 2]$ , so that  $\tilde{y}_1 = y_4$ ,  $\tilde{y}_2 = y_1$ ,  $\tilde{y}_3 = y_3$  and  $\tilde{y}_4 = y_2$ .



**Fig. 2.** Considering  $K_j = 2$  for all  $j$ , this figure depicts an example of the  $\prod_{j=1}^L K_j = 2^L = 8$  paths ( $L = 3$ ) of sequence of labels  $\tilde{y}_j$ ,  $j = 1, \dots, L$ . The best path with probability 0.2160 is shown with dashed lines.

Unfortunately, the optimal solution of these two problems is not feasible for anything but very small values of  $L$ ; the total space is  $(\prod_{j=1}^L K_j) \times L!$ . Hence, in this section we introduce an efficient double Monte Carlo strategy for quasi-optimal inference in Classifier Chains. We present both a tractable label prediction scheme at test time (MCC) and a method that performs an additional search for the optimal chain sequence at build time (M<sub>2</sub>CC).

### 3.1. Training step: finding the best chain

In order to obtain the best chain (i.e., the optimal label order) during the training step we introduce a *payoff function*<sup>4</sup>,

$$J(\mathbf{s}) = \sum_{i=1}^N p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, \mathbf{s}), \quad (8)$$

and the optimal sequence,  $\hat{\mathbf{s}}$ , is the one that maximizes (8) over the set of  $L!$  possible sequences, i.e.,

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmax}} J(\mathbf{s}) = \underset{\mathbf{s}}{\operatorname{argmax}} \sum_{i=1}^N p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, \mathbf{s}). \quad (9)$$

<sup>4</sup>This is an intuitive payoff function but, clearly, there are other possibilities. Note that  $\frac{1}{N} J(\mathbf{s})$  can be seen as a Monte Carlo approximation of the function  $\bar{J}(\mathbf{s}) = \sum_{i=1}^L \int p(\mathbf{y}_i | \mathbf{x}, \mathbf{s}) d\mathbf{x}$ .

The exact solution of (9) is intractable even for medium values of  $L$ . Therefore, we propose using the Monte Carlo approach summarized in Algorithm 1 to perform an efficient exploration of the label-sequence space. This algorithm starts with a randomly chosen label sequence,  $\mathbf{s}_0$ , which is then modified trying to find local maximum of the payoff function at least.

#### 3.1.1. Basic proposal procedure

In order to explore the space of  $\mathbf{s}$ , we require a proposal function. In this first algorithm we use the simplest possible proposal mechanism. Specifically, given a sequence

$$\mathbf{s}_{t-1} = [\mathbf{s}_{t-1}(1) = s_1, \dots, \mathbf{s}_{t-1}(L) = s_L]$$

the proposal function  $\pi(\mathbf{s}_t | \mathbf{s}_{t-1})$  consists of choosing uniformly two positions of the label sequence ( $1 \leq \ell, m \leq L$ ) and swapping the labels corresponding to those positions, so that  $\mathbf{s}_t(\ell) = \mathbf{s}_{t-1}(m)$  and  $\mathbf{s}_t(m) = \mathbf{s}_{t-1}(\ell - 1)$ .

### 3.2. Inference (test) step: finding the best path $\tilde{\mathbf{y}}^*$

In the test step, for a given test instance,  $\mathbf{x}^*$ , for which the true label association is unknown, and a label order (either estimated for M<sub>2</sub>CC or randomly chosen for MCC), we wish to

---

**Algorithm 1** Finding a suitable  $\hat{\mathbf{s}}$ 

Input:

- $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^D$ : training data
- $\pi(\mathbf{s}|\mathbf{s}_{t-1})$ : proposal density
- $T'$ : number of iterations

Algorithm:

1. Start with some random sequence,  $\mathbf{s}_0$ , and build an initial model,  $p(\mathbf{y}|\mathbf{x}, \mathbf{s}_0)$ .
2. For  $t = 1, \dots, T'$ :
  - (a) Draw  $\mathbf{s}' \sim \pi(\mathbf{s}|\mathbf{s}_{t-1})$  and build model  $p(\mathbf{y}|\mathbf{x}, \mathbf{s}')$ .
  - (b) if  $J(\mathbf{s}') > J(\mathbf{s}_{t-1})$ 
    - $\mathbf{s}_t \leftarrow \mathbf{s}'$  accept.
  - (c) else
    - $\mathbf{s}_t \leftarrow \mathbf{s}_{t-1}$  reject.

Output:

- $\hat{\mathbf{s}} = \mathbf{s}_{T'}$ : estimated label sequence.
- 

**Algorithm 2** Finding  $\mathbf{y}^*$  for a given test instance  $\mathbf{x}^*$ .

Input:

- $\mathbf{x}^*$ : test instance.
- $\hat{\mathbf{s}}$ : label order (estimated or chosen randomly).
- $p(\mathbf{y}|\mathbf{x}, \hat{\mathbf{s}})$ : probabilistic model (from training stage).

Algorithm:

1. Obtain an initial path,  $\mathbf{y}_0$ , using CC.
2. For  $t = 1, \dots, T$ :
  - (a) Draw  $\tilde{\mathbf{y}}' \sim p(\tilde{\mathbf{y}}|\mathbf{x}^*, \hat{\mathbf{s}})$
  - (b) if  $p(\tilde{\mathbf{y}}'|\mathbf{x}^*, \hat{\mathbf{s}}) > p(\tilde{\mathbf{y}}_t|\mathbf{x}^*, \hat{\mathbf{s}})$ 
    - $\tilde{\mathbf{y}}_t \leftarrow \tilde{\mathbf{y}}'$  accept.
  - (c) else
    - $\tilde{\mathbf{y}}_t \leftarrow \tilde{\mathbf{y}}_{t-1}$  reject.

Output:

- $\tilde{\mathbf{y}}^* = \tilde{\mathbf{y}}_T$ : predicted label assignment.
- 

find the optimal label vector that maximizes Eq. (7). In general, this problem can be solved analytically for low values of  $L$  by exploring all the  $\prod_{j=1}^L K_j$  possible paths, as in the PCC method [7]. However, when  $L$  grows this method quickly becomes computationally intractable. Therefore, we propose here using the random search Monte Carlo approach shown in Algorithm 2 to approximate Eq. (7). This algorithm starts from the greedy inference offered by standard CC, draws samples  $\mathbf{y}^{(i)}$ ,  $i = 1, \dots, T$  according to the model  $p(\tilde{\mathbf{y}}_t^*|\mathbf{x}^*, \hat{\mathbf{s}})$ , and provides a predicted label sequence

$$\tilde{\mathbf{y}}^* = \operatorname{argmax} p(\tilde{\mathbf{y}}_t^*|\mathbf{x}^*, \hat{\mathbf{s}}), \quad (10)$$

where  $\tilde{\mathbf{y}}_t^*$  ( $1 \leq t \leq T$ ) are the accepted samples. Note that in Algorithm 2 the candidate vectors  $\mathbf{y}'$  are all drawn *directly* from our target pdf  $p(\mathbf{y}|\mathbf{x}^*, \hat{\mathbf{s}})$ . Indeed, it is easy to draw from, since  $\mathbf{y}'$  is a possible path on a tree (see Figure 2). From a Monte Carlo point of view, this is an important consideration that guarantees  $\mathbf{y}^* = \mathbf{y}_T$  will have (at least) a high probability  $p(\mathbf{y}_T|\mathbf{x}^*, \hat{\mathbf{s}})$ .

### 3.3. First two proposed methods: MCC and $M_2CC$

In order to compare fairly both the performance and the computational effort, we progressively apply the ideas introduced in this section. We define two methods:

- MCC: given a classifier chain trained on some previously-determined sequence of labels  $\mathbf{s}$  (e.g., randomly as in CC, PCC), will infer label sets for all test instances; essentially a tractable version of PCC, using Algorithm 2.
- $M_2CC$ : in the training step, additionally searches for a suitable sequence of labels  $\hat{\mathbf{s}}$ , namely we also use Algorithm 1. Therefore, Algorithm 1 and Algorithm 2 jointly outline the entire method  $M_2CC$ .

## 4. ENHANCEMENTS

### 4.1. Population of label orders

Since the payoff function  $J(\mathbf{s})$  uses  $p(\mathbf{y}|\mathbf{x}, \mathbf{s})$  that is just an approximation of the true data distribution,  $f(\mathbf{y}|\mathbf{x}, \mathbf{s})$ , and in order to decrease the dependence to the training step (then to diminish the overfitting), we can consider a population of label orders  $\mathcal{S} = \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}\}$ . Namely, we can generate a random walk in the label order space  $\{1, \dots, L\}^L$  (using a suitable proposal density  $\pi(\mathbf{s}|\mathbf{s}_{t-1})$ ) and, after  $T'$  iterations, take the best  $M$  label orders  $\mathbf{s}^{(i)}$  in terms of greater payoff function  $J(\mathbf{s}^{(i)})$ . The method is detailed in Algorithm 3.

---

**Algorithm 3** Finding a suitable population  $\mathcal{S} = \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}\}$ 


---

Input:

- $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^D$ : training data
- $\pi(\mathbf{s}|\mathbf{s}_{t-1})$ : proposal density
- $T'$ : number of iterations
- $M$ : number of output sequences

Algorithm:

1. Start with some random sequence  $\mathbf{s}_0$
2. For  $t = 1, \dots, T'$ :
  - (a)  $\mathbf{s}' \sim \pi(\cdot|\mathbf{s}_{t-1})$
  - (b) if  $J(\mathbf{s}') \geq J(\mathbf{s}_t)$ 
    - $\mathbf{s}_t \leftarrow \mathbf{s}'$  accept
    - $w_t \leftarrow J(\mathbf{s}')$  set
  - (c) Otherwise, if  $J(\mathbf{s}') < J(\mathbf{s}_t)$ 
    - $\mathbf{s}_t \leftarrow \mathbf{s}_{t-1}$  accept
    - $w_t \leftarrow J(\mathbf{s}_{t-1})$  set
3. Sort  $\mathbf{s}_1, \dots, \mathbf{s}_{T'}$  in decreasing order according to  $w_1, \dots, w_{T'}$  and select the first  $M$  sequence,  $\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}$ .

Output:

- $\mathcal{S} = \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}\}$ .
  - corresponding weights  $w^{(1)}, \dots, w^{(M)}$ .
- 

Then, in the test step, we can use the information provided by the population  $\mathcal{S} = \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}\}$ . The trivial

way, is to run different parallel algorithms to find sequences of labels  $\tilde{\mathbf{y}}^{(i)}$  using different  $s^{(i)}$ ,  $i = 1, \dots, M$ , and then taking the best one. However, we propose a more sophisticated procedure running only one random search to find a suitable  $\tilde{\mathbf{y}}^*$  using the entire information in the population  $\mathcal{S}$ . The corresponding idea is shown in Algorithm 4.

---

**Algorithm 4** Finding  $\tilde{\mathbf{y}}^*$  given a  $\mathbf{x}^*$  and a population  $\mathcal{S}$ .

---

Input:

- $\mathbf{x}^*$  a test instance
- $M$  sequences  $\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}$  (obtained in the training step)
- $M$  weights  $w^{(1)}, \dots, w^{(M)}$  (obtained in the training step)
- $p(\mathbf{y}|\mathbf{x}, \mathbf{s})$  (obtained in the training step)

Algorithm:

1.  $\mathbf{y}_0$  an initial path using CC's greedy inference
2. For  $t_1 = 1, \dots, T_1$ :
  - (a) Choose a  $\mathbf{s}_{t_1} \in \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(M)}\}$  according (proportionally) to the weights  $w^{(j)}$ ,  $j = 1, \dots, M$ .
  - (b) Set  $\mathbf{z}_1 = \mathbf{y}_{t_1-1}$ .
  - (c) For  $t_2 = 1, \dots, T_2$ :
    - i.  $\mathbf{z}' \sim p(\mathbf{z}|\mathbf{x}^*, \mathbf{s}_{t_1})$
    - ii. if  $p(\mathbf{z}'|\mathbf{x}^*, \mathbf{s}_{t_1}) > p(\mathbf{z}_{t_2}|\mathbf{x}^*, \mathbf{s}_{t_1})$ 
      - $\mathbf{z}_{t_2+1} \leftarrow \mathbf{z}'$  accept
    - iii. else
      - $\mathbf{z}_{t_2+1} \leftarrow \mathbf{z}_{t_2}$  reject
  - (d) Set  $\mathbf{y}_{t_1} = \mathbf{z}_{T_2}$ .

Output:

- $\tilde{\mathbf{y}}^* = \mathbf{y}_{T_1}$
- 

## 4.2. Improved proposal density

Searching the sequence space requires building a classifier for each sequence we want to try, and is thus inherently much more expensive than searching the label space. This cost restricts how many sequences we can look at. Furthermore, we note that changing the initial ‘links’ in the chain (i.e.,  $s_1, s_2, \dots$ ) implies a larger jump in the space than changing the final links (i.e.,  $\dots, s_{L-1}, s_L$ ), due to the chain structure.

In light of these observations, we propose an improvement of the proposal  $\pi(\mathbf{s}|\mathbf{s}_{t-1})$  that tries a new sequence  $\mathbf{s}'$  based on the previous; where the links in the chain are progressively frozen from beginning to end. Building  $\mathbf{s}'$  implies a complexity of  $L$  (classifiers to train). However, given a sequence  $\mathbf{s}$  which we have already built the complexity is  $L - \ell$  where  $\ell$  is the highest number where the following holds:

$$\sum_{j=1}^{\ell} I(s_j = s'_j) = \ell$$

where  $1 \leq \ell \leq L$ . Thus, by freezing the initial links, the probability of  $\ell \rightarrow L$  increases, and thus the sequence is cheaper

to build, and the sequence space cheaper to explore. We do this using the current iteration  $t$  as the temperature.

Algorithm 5 details our new proposal density, which gradually freezes the sequence over time.

---

**Algorithm 5** Improved proposal density  $\pi(\mathbf{s}|\mathbf{s}_{t-1})$ 


---

Input:

- $\mathbf{s}_{t-1}$ : previous label order, i.e.,  $\mathbf{s}_{t-1}$  is a sequence  $\mathbf{s}_{t-1} = [s_1, \dots, s_L]$  of  $L$  indices, at the iteration  $t-1$
- $t'$ : a time parameter

Proposal procedure:

1. Choose a value  $\mathbf{s}_{t-1}(i) = s_i$  where  $i \in \{1, \dots, L\}$  according to the with probability

$$p_{i,t} \propto \begin{cases} \frac{1}{N}, & t < t', \\ \left(\frac{1}{N}\right)^{\frac{\beta t}{i}}, & t \geq t'. \end{cases} \quad (11)$$

where  $\beta > 0$  is a constant.

2. Choose a value  $\mathbf{s}_{t-1}(k) = s_k$  where  $k \neq i$  according to the with probability

$$p_{k,t} \propto \begin{cases} \frac{1}{N-1}, & t < t', \\ \left(\frac{1}{N-1}\right)^{\frac{\beta t}{k}}, & t \geq t'. \end{cases} \quad (12)$$

3. Set  $\mathbf{s}_t = \mathbf{s}_{t-1}$  and then set  $\mathbf{s}_t(k) = \mathbf{s}_{t-1}(i)$  and  $\mathbf{s}_t(i) = \mathbf{s}_{t-1}(k)$ .

Output:

- $\mathbf{s}_t$ .
- 

## 4.3. Enhanced Methods $\text{EM}_2\text{CC}$ and $\text{E}_e\text{M}_2\text{CC}$

We have already described the MCC and the  $\text{M}_2\text{CC}$ . Now we also define other techniques:

- $\text{EM}_2\text{CC}$ : In this method, we use both the Algorithm 3 and the Algorithm 4, jointly. However, we still use the trivial proposal density  $\pi(\mathbf{s}|\mathbf{s}_{t-1})$  described in Section 3.1.1.
- $\text{E}_e\text{M}_2\text{CC}$ : In this case, another time we use both the Algorithm 3 and the Algorithm 4 and the improved proposal  $\pi(\mathbf{s}|\mathbf{s}_{t-1})$  described in Algorithm 5.

## 5. EXPERIMENTS

We perform experiments on a collection of real world datasets familiar to the multi-label literature [6, 5, 7], e.g., see Table 2.

We compare to baseline  $\text{IC}$  [9], the original classifier chains method  $\text{CC}$  [6], the Bayes-optimal rendition  $\text{PCC}$  [7]; and also the conditional dependency networks method  $\text{CDN}$  of [8] under  $I = 1000$  total iterations. See Section 2 and the references therein for more information on these methods. For our methods MCC and  $\text{M}_2\text{CC}$ , we use  $T = 100$  (inference

$y$ -step) and just  $T' = 10$  for  $M_2CC$  (training  $s$ -step). Clearly, better results can be easily obtained increasing  $T'$ .

As a base classifier we use *Support Vector Machines* (SVMs) fitted with logistic models (as according to [4]). Logistic Regression has been a popular choice in the probabilistic multi-label literature [7, 8] due to its probabilistic output. However, we have found that SVM-based methods perform much better in comparison. By default an SVM will not provide probabilistic output (i.e.,  $h_j(\mathbf{x}) \in [0, 1]$ ) and for this reason we fit the logistic models.

All methods are implemented and will be made available within the MEKA framework<sup>5</sup>; an open-source framework based on the WEKA machine learning framework [12] with added support for multi-label classification and evaluation.

### 5.1. Evaluation Measures

Multi-label evaluation measures can be categorised as either *label-based* measures that evaluate the performance of each label individually (and take an average across all labels); or *example-based* measures that evaluate the performance of label-classification vectors as if each were a single class value [2]. Of the latter, we use the standard *exact match* measure where, given predictions  $\mathbf{y}^*$  and ground truths  $\mathbf{y}$ :

$$\text{EXACT MATCH} = \frac{1}{N} \sum_{i=1}^N I(\mathbf{y}^{(i)} = \mathbf{y}^{*(i)})$$

To contrast with this measure, we use the well-known label-based *Hamming score*<sup>6</sup>:

$$\text{HAMMING SCORE} = \frac{1}{NL} \sum_{i=1}^N \sum_{j=1}^L I(y_j^{(i)} = \hat{y}_j^{(i)})$$

Note that  $I(\cdot)$  is simply an identity function in both cases.

### 5.2. Results

We carry out 5-fold cross validation. Results for predictive performance are displayed in Table 3. Results for running time performance are given in Table 4.

We have also provided the ranks and average ranks of each method and significance results according to the Nemenyi test [13];  $a \succ b$  indicates that algorithm  $a$  is significantly better than  $b$  (under a  $p$ -value of 0.10).

The original CC paper [6] also presented CC in Bagging ensembles (ECC) to improve predictive performance. We additionally compare to this method.

### 5.3. Discussion

- As claimed in the literature, CC improves over IC considerable in EXACT MATCH and is similar under HAMMING SCORE.
- PCC in turn improves on CC – that is, in the cases where it is tractable – also across all evaluation measures.
- Our MCC method outperforms CC on almost every occasion.
- MCC is identical to PCC on all datasets that it finishes on, but is much more tractable than PCC.
- M<sub>2</sub>CC obtains similar performance to MCC on these datasets overall.
- EM<sub>2</sub>CC obtains best performance of all methods.
- E<sub>e</sub>M<sub>2</sub>CC obtains performance almost as good as EM<sub>2</sub>CC, but is more efficient on most datasets. It is not more efficient on some of the large datasets, indicating that there may be some overhead in this particular implementation sensitive to  $L$ .
- Our methods are generally faster than CDN, especially for larger  $L$ .
- ECC improves over CC, but even these 10 CC models are unable to compete with our enhanced methods.
- There is clearly a qualitative difference between the multi-dimensional datasets, and the sub-problems of binary multi-label datasets.

## 6. CONCLUSIONS AND FUTURE WORK

We introduced a novel double Monte Carlo technique for multi-dimensional learning using classifier chains. A Monte Carlo search technique is used both to efficiently search the label-path space at inference time and also the chain-sequence space at training time. We show with an extensive empirical evaluation that using our techniques results in better predictive performance than related methods while remaining computationally tractable. Our model convincingly obtains overall best predictive performance of all the methods we looked at, and proves tractable enough for real-world applications.

In future work, we intend to look at more advanced random search algorithms and dependency structures other than chain models (see Eq. (5)), as well different payoff functions to evaluate the sequence of labels.

<sup>5</sup><http://meka.sourceforge.net>

<sup>6</sup>Often posed as *Hamming loss*; i.e.,  $1 - \text{Hamming score}$

**Table 1.** The methods we consider and their parameters. The novel methods we present are below the middle line; where each inherits the parameters of the previous, e.g.,  $E_eM_2CC$  takes parameters  $T = 100, T' = 50, M = 10, \beta = 0.03$ . For CDN,  $T_c$  is the number of collection iterations. For ECC,  $M$  is the number of models.

Key	Method	Parameters	Reference
IC	Independent Classifiers		[9]
CC	Classifier Chains		[6]
ECC	Ensembles of Classifier Chains	$M = 10$	[6]
PCC	Probabilistic Classifier Chains		[7]
CDN	Conditional Dependency Networks	$T = 1000, T_c = 100$	[8]
MCC	Monte Carlo Optimization (y-space) for CC	$T = 100$	Algorithm 1
$M_2CC$	2x Monte Carlo Optimization (y-space, s-space) for CC	$T' = 50$	Algorithm 2
$EM_2CC$	Enhanced $M_2CC$ (with s-populations) for CC	$M = 10$	Algorithm 5
$E_eM_2CC$	Enhanced $M_2CC$ (efficient populations) for CC	$\beta = 0.03$	Algorithm 3

**Table 2.** A collection of datasets and associated statistics, where LC is *label cardinality*: the average number of labels relevant to each example; relevant for binary labels. We have divided multi-dimensional datasets, and multi-label (binary-only) datasets.

	$N$	$L$	$K$	$d$	LC	Type
Solar Flare	323	3	5	10	N/A	astrology
Bridges	107	5	2–6	7	N/A	civil engineering
Thyroid	9172	7	2–5	28	N/A	medical
Parkinson's	488	5	3	58	N/A	medical
Music	593	6	2	72	1.87	audio
Scene	2407	6	2	294	1.07	image
Yeast	2417	14	2	103	4.24	biology
Genbase	661	27	2	1185	1.25	biology
Medical	978	45	2	1449	1.25	medical/text
Enron	1702	53	2	1001	3.38	email/text
Reuters	6000	103	2	500	1.46	news/text

## 7. REFERENCES

- [1] Jesse Read, *Scalable Multi-label Classification*, Ph.D. thesis, University of Waikato, 2010.
- [2] G. Tsoumakas, I. Katakis, and I. Vlahavas, “Mining multi-label data,” in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. 2010, 2nd edition, Springer.
- [3] Julio H. Zaragoza, Luis Enrique Sucar, Eduardo F. Morales, Concha Bielza, and Pedro Larrañaga, “Bayesian chain classifiers for multidimensional classification,” in *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI '11)*, 2011.
- [4] Trevor Hastie and Robert Tibshirani, “Classification by pairwise coupling,” in *Advances in Neural Information Processing Systems*, Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, Eds. 1998, vol. 10, MIT Press.
- [5] Grigorios Tsoumakas and Ioannis P. Vlahavas, “Random k-labelsets: An ensemble method for multilabel classification,” in *ECML '07: 18th European Conference on Machine Learning*. 2007, pp. 406–417, Springer.
- [6] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank, “Classifier chains for multi-label classification,” *Machine Learning*, vol. 85, no. 3, pp. 333–359, 2011.
- [7] Weiwei Cheng, Krzysztof Dembczyński, and Eyke Hüllermeier, “Bayes optimal multilabel classification via probabilistic classifier chains,” in *ICML '10: 27th International Conference on Machine Learning*, Haifa, Israel, June 2010, Omnipress.
- [8] Yuhong Guo and Suicheng Gu, “Multi-label classification using conditional dependency networks.,” in *IJCAI '11: Proceedings of the 24th International Conference on Artificial Intelligence*. 2011, pp. 1300–1305, IJCAI/AAAI.

**Table 3.** Predictive Performance from 5-fold CV, displayed as: value (rank), i.e., the average value across all folds and the rank of that value for each dataset.

Dataset	EXACT MATCH								
	IC	CC	PCC	ECC	CDN	MCC	M <sub>2</sub> CC	EM <sub>2</sub> CC	E <sub>e</sub> M <sub>2</sub> CC
SolFlare	0.774 (7)	0.796 (1)	0.780 (5)	0.690 (8)	0.591 (9)	0.780 (5)	0.786 (2)	0.786 (2)	0.786 (2)
Bridges	0.094 (9)	0.122 (4)	0.122 (4)	0.103 (8)	0.140 (1)	0.122 (4)	0.130 (2)	0.130 (2)	0.111 (7)
Parkins	0.174 (1)	0.172 (2)	0.164 (5)	0.162 (8)	0.164 (5)	0.164 (5)	0.162 (8)	0.170 (4)	0.172 (2)
Thyroid	0.835 (6)	0.016 (9)	0.842 (3)	0.820 (7)	0.783 (8)	0.842 (3)	0.842 (3)	0.845 (2)	0.846 (1)
Music	0.299 (7)	0.287 (9)	0.346 (4)	0.314 (6)	0.297 (8)	0.346 (4)	0.358 (3)	0.370 (1)	0.367 (2)
Scene	0.538 (8)	0.545 (7)	0.636 (3)	0.608 (6)	0.531 (9)	0.636 (3)	0.632 (5)	0.677 (2)	0.685 (1)
Yeast	0.140 (7)	0.151 (6)	DNF	0.186 (5)	0.069 (8)	0.209 (4)	0.220 (3)	0.225 (1)	0.223 (2)
Genbase	0.941 (8)	0.964 (2)	DNF	0.945 (6)	0.945 (6)	0.964 (2)	0.964 (2)	0.962 (5)	0.965 (1)
Medical	0.585 (8)	0.622 (4)	DNF	0.643 (1)	0.602 (7)	0.629 (2)	0.625 (3)	0.621 (5)	0.604 (6)
Enron	0.065 (8)	0.099 (3)	DNF	0.112 (1)	0.073 (7)	0.101 (2)	0.087 (6)	0.093 (5)	0.096 (4)
Reuters	0.287 (7)	0.346 (6)	DNF	0.364 (5)	0.271 (8)	0.366 (4)	0.371 (1)	0.371 (1)	0.367 (3)
avg. rank	6.91	4.82	4.00	5.55	6.91	3.45	3.45	2.73	2.82

Nemenyi significance: MCC > IC; MCC > CDN; M<sub>2</sub>CC > IC; M<sub>2</sub>CC > CDN; EM<sub>2</sub>CC > IC; EM<sub>2</sub>CC > CDN; E<sub>e</sub>M<sub>2</sub>CC > IC; E<sub>e</sub>M<sub>2</sub>CC > CDN;

Dataset	HAMMING SCORE								
	IC	CC	PCC	ECC	CDN	MCC	M <sub>2</sub> CC	EM <sub>2</sub> CC	E <sub>e</sub> M <sub>2</sub> CC
SolFlare	0.901 (7)	0.916 (1)	0.903 (4)	0.846 (8)	0.772 (9)	0.903 (4)	0.902 (6)	0.904 (2)	0.904 (2)
Bridges	0.634 (7)	0.664 (3)	0.666 (1)	0.635 (6)	0.616 (9)	0.666 (1)	0.650 (4)	0.650 (4)	0.633 (8)
Parkins	0.681 (2)	0.684 (1)	0.673 (7)	0.679 (3)	0.679 (3)	0.673 (7)	0.673 (7)	0.679 (3)	0.678 (6)
Thyroid	0.974 (1)	0.829 (9)	0.974 (1)	0.970 (7)	0.961 (8)	0.974 (1)	0.973 (6)	0.974 (1)	0.974 (1)
Music	0.808 (4)	0.789 (8)	0.802 (6)	0.805 (5)	0.788 (9)	0.802 (6)	0.810 (3)	0.811 (1)	0.811 (1)
Scene	0.891 (7)	0.865 (8)	0.894 (4)	0.897 (3)	0.857 (9)	0.894 (4)	0.892 (6)	0.904 (2)	0.907 (1)
Yeast	0.790 (1)	0.752 (7)	DNF	0.788 (3)	0.718 (8)	0.783 (6)	0.787 (4)	0.789 (2)	0.787 (4)
Genbase	0.997 (7)	0.999 (1)	DNF	0.997 (7)	0.998 (5)	0.999 (1)	0.999 (1)	0.998 (5)	0.999 (1)
Medical	0.987 (5)	0.988 (2)	DNF	0.989 (1)	0.986 (8)	0.988 (2)	0.988 (2)	0.987 (5)	0.987 (5)
Enron	0.925 (2)	0.922 (5)	DNF	0.939 (1)	0.924 (3)	0.922 (5)	0.921 (8)	0.922 (5)	0.924 (3)
Reuters	0.985 (1)	0.985 (1)	DNF	0.985 (1)	0.983 (8)	0.985 (1)	0.985 (1)	0.985 (1)	0.985 (1)
avg. rank	4.00	4.18	3.83	4.09	7.18	3.45	4.36	2.82	3.00

Nemenyi significance: PCC > CDN; MCC > CDN; EM<sub>2</sub>CC > CDN; E<sub>e</sub>M<sub>2</sub>CC > CDN;

**Table 4.** Running time performance (in seconds); average of 5-fold CV.

Dataset	IC	CC	PCC	ECC	CDN	MCC	M <sub>2</sub> CC	EM <sub>2</sub> CC	E <sub>e</sub> M <sub>2</sub> CC
SolFlare	0 (9)	0 (8)	1 (6)	3 (5)	7 (4)	1 (7)	13 (1)	7 (3)	7 (2)
Bridges	1 (8)	0 (9)	2 (6)	5 (5)	8 (4)	1 (7)	22 (1)	13 (2)	12 (3)
Parkins	2 (9)	2 (8)	5 (6)	9 (5)	19 (4)	4 (7)	80 (1)	50 (2)	44 (3)
Thyroid	28 (9)	31 (8)	820 (4)	126 (6)	208 (5)	45 (7)	1417 (1)	1004 (2)	900 (3)
Music	0 (9)	0 (8)	1 (7)	2 (6)	6 (4)	5 (5)	45 (1)	18 (2)	10 (3)
Scene	12 (8)	11 (9)	15 (7)	44 (6)	92 (4)	90 (5)	1347 (1)	684 (2)	335 (3)
Yeast	11 (8)	11 (7)	DNF	66 (6)	88 (5)	149 (4)	1313 (1)	731 (2)	546 (3)
Genbase	11 (7)	8 (8)	DNF	56 (6)	573 (5)	1695 (2)	5287 (1)	774 (4)	823 (3)
Medical	9 (8)	11 (7)	DNF	86 (6)	1546 (3)	3420 (2)	6940 (1)	1038 (5)	1192 (4)
Enron	102 (7)	92 (8)	DNF	349 (6)	3091 (4)	3884 (2)	10821 (1)	2986 (5)	3470 (3)
Reuters	106 (8)	120 (7)	DNF	20593 (1)	14735 (3)	1837 (2)	5740 (4)	4890 (6)	5310 (5)
avg. rank	8.18	7.91	6.00	5.27	4.09	4.55	1.27	3.18	3.18

- [9] Grigorios Tsoumakas and Ioannis Katakis, “Multi label classification: An overview,” *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.
- [10] Jesse Read, Bernhard Pfahringer, and Geoff Holmes, “Multi-label classification using ensembles of pruned sets,” in *ICDM’08: Eighth IEEE International Conference on Data Mining*. 2008, pp. 995–1000, IEEE.
- [11] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Deroski, “An extensive experimental comparison of methods for multi-label learning,” *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, Sept. 2012.
- [12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Reutemann Peter, and Ian H. Witten, “The weka data mining software: An update,” *SIGKDD Explorations*, vol. 11, no. 1, 2009.
- [13] Janez Demšar, “Statistical comparisons of classifiers over multiple data sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.